Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 452602, 9 pages
http://dx.doi.org/10.1155/2015/452602

Research Article

Note on a Single-Machine Scheduling Problem with Sum of Processing Times Based Learning and Ready Times

Shang-Chia Liu,[1] Wei-Ling Hung,[2] and Chin-Chia Wu[2]

[1]*Department of Business Administration, Fu Jen Catholic University, New Taipei City 24205, Taiwan*
[2]*Department of Statistics, Feng Chia University, Taichung 40724, Taiwan*

Correspondence should be addressed to Shang-Chia Liu; 056298@mail.fju.edu.tw

Received 5 September 2014; Accepted 20 January 2015

Academic Editor: Chuangxia Huang

In the recent 20 years, scheduling with learning effect has received considerable attention. However, considering the learning effect along with release time is limited. In light of these observations, in this paper, we investigate a single-machine problem with sum of processing times based learning and ready times where the objective is to minimize the makespan. For solving this problem, we build a branch-and-bound algorithm and a heuristic algorithm for the optimal solution and near-optimal solution, respectively. The computational experiments indicate that the branch-and-bound algorithm can perform well the problem instances up to 24 jobs in terms of CPU time and node numbers, and the average error percentage of the proposed heuristic algorithm is less than 0.5%.

1. Introduction

In the classical scheduling models, most researchers consider that the job processing times are all constant numbers. In fact, it can be seen in many real situations that a production time can be shortened if it is operated later due to the fact that the efficiency of the production facility (e.g., a machine or a worker) continuously improves with time. This situation is named as the "*learning effect*" in the research community [1, 2]. Furthermore, Biskup [3] provided a survey paper to discuss different learning models in the scheduling research.

More recently, the *learning effect* has continued to receive a lot of effort. For more recent problems with time-dependent processing times on single-machine settings, we refer readers to Cheng et al. [4], Eren and Güner [5, 6], Eren [7], Janaik and Rudek [8], Toksarı et al. [9], Toksari and Güner [10], Wang and Liu [11], Wang et al. [12], Yin et al. [13], Yin et al. [14–17], Wu et al. [18–20], Wang et al. [21–23], Bai et al. [24], Vahedi-Nouri et al. [25], Lu et al. [26], and so forth.

Meanwhile, the concept of the learning process with ready times is relatively limited. For example, Bachman and Janiak [27] showed that the makespan single-machine job position-based learning problem is NP-hard in the strong sense. Following the same model by Bachman and Janiak [27], Lee et al. [28] built exact and heuristic algorithms to solve the optimal and near-optimal solutions, respectively. Eren [29] formulated a nonlinear mathematical programming model for the single-machine learning scheduling problem with different ready times. Lee et al. [30] explore a single-machine position-based learning scheduling problem with ready times to minimize the sum of makespan and total completion time. Wu and Liu [31] dealt with a single-machine problem with the learning effect and release times where the objective is to minimize the makespan. They proposed a branch-and-bound algorithm and three two-stage heuristic algorithms for the problem. Wu et al. [32] considered a single-machine problem with the sum of processing times based learning effect and release times to minimize the total completion times. They developed a branch-and-bound algorithm for the optimal solution. Then they proposed a simulated-annealing heuristic algorithm for a near-optimal solution. Wu et al. [33] considered a single-machine problem with learning effect and ready times where the objective is to minimize the total weighted completion time. They developed a branch-and-bound algorithm and a simulated annealing algorithm for the problem. Wu et al. [34] considered a single-machine problem with the sum of processing time based learning effect and release times to minimize the total weighted completion times. They

developed a branch-and-bound algorithm incorporating with several dominance properties and two lower bounds for the optimal solution and then used a genetic heuristic-based algorithm for a near-optimal solution. J.-B. Wang and J.-J. Wang [35] investigated a single-machine scheduling problem with time-dependent processing times and group technology assumption to minimize the makespan with ready times of the jobs. They showed that the problem can be solved in polynomial time when starting time-dependent processing times and group technology simultaneously.

Besides, Dessouky [36] pointed out the importance of ready time in semiconductor manufacturing where it is common to find newer, more modern machines running side by side with older, less efficient machines which are kept in operation because of high replacement cost. Moreover, all of the above works deal with job position-based learning. Following Kuo and Yang [37] model, in this paper, we explored the jobs with different release times into the sum of processing time based learning mode.

The branch-and-bound method is a general algorithm for finding optimal solutions of various optimization problems. However, the execution time required is impractical when the number of activities increases. Therefore, a branch-and-bound algorithm usually incorporates with dominance properties and lower bounds to derive the optimal solution (Lee et al. [28, 30]). Thus, we also developed a branch-and-bound algorithm incorporating with several dominances and two lower bounds to derive the optimal solution.

This paper is organized as follows. In Section 2, we define the problem formulation. In Section 3, we derive some dominance properties and two lower bounds used in the branch-and-bound method and state the descriptions of the proposed heuristic. In Section 4, we report the results of a computational experiment. Conclusions could be found in the last section.

2. Problem Statement and Notation Definition

Below are stated some notations used throughout the paper.

n denotes the size number of given jobs.

S, S' denote the sequences of n jobs.

p_i denotes the basic processing time of job i.

r_i denotes the ready time of job i.

$C_i(S)$ is defined as the completion time of job i scheduled in a sequence S.

a denotes the learning effect with $a < 0$.

$p_{[i]}(S)$ denotes the basic processing time for the job scheduled in the ith position in S.

$r_{[i]}(S)$ denotes the ready time of a job scheduled in the ith position in S.

$C_{[i]}(S)$ denotes the completion time of a job scheduled in the ith position in S.

$p_{(1)} \le p_{(2)} \le \cdots \le p_{(n)}$ denotes a nondecreasing order of processing times $p_1, p_2, \ldots, p_n$.

$r_{(1)} \le r_{(2)} \le \cdots \le r_{(n)}$ denotes a nondecreasing order of ready times $r_1, r_2, \ldots, r_n$.

At first, we assume that there are n given jobs to be operated on a single machine. Let each job j have its processing time p_j and a ready time r_j. Due to the learning effect, we assume that the actual processing time of job j is $p_{jr} = p_j(1 + \sum_{l=1}^{r-1} p_{[l]})^a$ if it is scheduled in the rth position where $a < 0$ is a learning ratio common for all jobs. The objective of this paper is to minimize the makespan of n given jobs in a sequence.

3. Solution Methods

Due to the fact that the proposed problem is a difficult one, we attempt to use a branch-and-bound method and a heuristic algorithm for this problem. In order to speed up the searching process, we derive several dominance propositions and two lower bounds used in the branch-and-bound method.

Next, we derive five properties based on a pairwise interchange of two adjacent jobs. The proofs of Propositions 2 to 5 are omitted since they are similar to that of Proposition 1.

Proposition 1. *If $p_i < p_j$ and $\max\{r_i, r_j\} \le A$, then there is an optimal schedule in which job i is scheduled before job j.*

Proof. Consider two sequences $S = (\pi, J_i, J_j, \pi')$ and $S' = (\pi, J_j, J_i, \pi')$, where π and π' denote partial sequences. To show that S dominates S', it suffices to show that $C_j(S) - C_i(S') < 0$. In addition, let A be the completion time of the last job in the subsequence π. Since $\max\{r_i, r_j\} \le A$, we have

$$
\begin{aligned}
C_i(S) &= A + p_i\left(1 + \sum_{l=1}^{r-1} p_{[l]}\right)^a,\\
C_j(S) &= A + p_i\left(1 + \sum_{l=1}^{r-1} p_{[l]}\right)^a + p_j\left(1 + \sum_{l=1}^{r-1} p_{[l]} + p_i\right)^a,\\
C_j(S') &= A + p_j\left(1 + \sum_{l=1}^{r-1} p_{[l]}\right)^a,
\end{aligned}
\tag{1}
$$

$$
C_i(S') = A + p_j\left(1 + \sum_{l=1}^{r-1} p_{[l]}\right)^a + p_i\left(1 + \sum_{l=1}^{r-1} p_{[l]} + p_j\right)^a. \tag{2}
$$

After taking the difference of (1) and (2), we have

$$C_i\left(S'\right) - C_j(S) = \left(p_j - p_i\right)\left(1 + \sum_{l=1}^{r-1} p_{[l]}\right)^a + p_i\left(1 + \sum_{l=1}^{r-1} p_{[l]} + p_j\right)^a - p_j\left(1 + \sum_{l=1}^{r-1} p_{[l]} + p_i\right)^a. \tag{3}$$

On substituting $\lambda = p_j/p_i$, $v = (1 + \sum_{l=1}^{r-1} p_{[l]})$, and $x = (p_i/\sum_{l=1}^{n} p_l)$ into (3) and simplifying, we obtain

$$C_i\left(S'\right) - C_j(S) = p_i v^a\left[(\lambda - 1) + (1 + \lambda x)^a - \lambda(1 + x)^a\right]. \tag{4}$$

Equation (4) can be easily obtained by taking the first and the second derivatives to λ for $\lambda \geq 1$, $a \leq 0$, and $x > 0$, and we have $C_i(S') - C_j(S) > 0$. Therefore, S dominates S'. □

Proposition 2. *If $r_i \leq A \leq r_j \leq A + p_i(1 + \sum_{l=1}^{r-1} p_{[l]})^a$ and $p_i < p_j$, then S dominates S'.*

Proposition 3. *If $A \geq r_i$ and $A + p_i(1 + \sum_{l=1}^{r-1} p_{[l]})^a < r_j$, then there is an optimal schedule in which job i is scheduled before job j.*

Proposition 4. *If $A \leq r_i \leq r_j$, $r_i + p_i(1 + \sum_{l=1}^{r-1} p_{[l]})^a \geq r_j$, and $p_i < p_j$, then there is an optimal schedule in which job i is scheduled before job j.*

Proposition 5. *If $A \leq r_i$ and $r_i + p_i(1 + \sum_{l=1}^{r-1} p_{[l]})^a \geq r_j$, then there is an optimal schedule in which job i is scheduled before job j.*

In what follows, two more properties to determine the ordering of the remaining unscheduled jobs are developed. Let $S = (\pi, \pi^c)$ and $S_1 = (\pi, \pi')$ be two sequences of jobs in which π denotes the scheduled part containing k jobs, π^c denotes the unscheduled part, and the jobs in π' are scheduled in the SPT rule; that is, $p_{(k+1)} \leq p_{(k+2)} \leq \cdots \leq p_{(n)}$.

Proposition 6. *If there exists job $j \in \pi^c$ such that $\max\{C_{[k-1]}(S), r_j\} + p_j(1 + \sum_{l=1}^{k} p_{[l]})^a < r_{[k]}$, then $S = (\pi, \pi^c)$ is a dominated sequence.*

Proposition 7. *If $C_{[k]}(S_1) > \max_{j\in\pi^c}\{r_j\}$, then $S_1 = (\pi, \pi')$ dominates sequences of the type (π, π^c) for any unscheduled sequence π^c.*

Following that, we will propose two simple lower bounds to curtail the branching tree. According to the definition, the completion time for the $(k+1)$th job is given by

$$C_{[k+1]}(S) = \max\left\{C_{[k]}(S), r_{[k+1]}\right\} + p_{[k+1]}\left(1 + \sum_{l=1}^{k} p_{[l]}\right)^a \geq C_{[k]}(S) + p_{[k+1]}\left(1 + \sum_{l=1}^{k} p_{[l]}\right)^a. \tag{5}$$

The makespan for S is easily obtained as follows:

$$C_{[n]}(S) \geq C_{[k]}(S) + \sum_{j=1}^{n-k} p_{[k+j]}\left(1 + \sum_{l=1}^{k} p_{[l]} + \sum_{l=1}^{j-1} p_{[k+l]}\right)^a. \tag{6}$$

According to (6), we have

$$\mathrm{lb}_1 = C_{[k]}(S) + \sum_{j=1}^{n-k} p_{(k+j)}\left(1 + \sum_{l=1}^{k} p_{[l]} + \sum_{l=1}^{j-1} p_{(n-k-l+1)}\right)^a, \tag{7}$$

where $p_{(k+1)} \leq p_{(k+2)} \leq \cdots \leq p_{(n)}$. In a similar way, we have

$$\mathrm{lb}_2 = \max_{1\leq j\leq n-k}\left\{r^*_{(k+j)} + \sum_{j=1}^{n-k} p^*_{(k+j)}\left(1 + \sum_{l=1}^{k} p_{[l]} + \sum_{l=1}^{j-1} p^*_{(n-k-l+1)}\right)^a\right\}, \tag{8}$$

where $p^*_{(k+1)} \leq p^*_{(k+2)} \leq \cdots \leq p^*_{(n)}$ denote the processing times of the unscheduled jobs arranged in a nondecreasing order.

In order to make the lower bound tighter, we choose the maximum value from (7) and (8) as the lower bound. That is,

$$\mathrm{lb} = \max\left\{\mathrm{lb}_1, \mathrm{lb}_2\right\}. \tag{9}$$

A typical approach to a NP-hard problem is to provide a heuristic algorithm. In what follows, a hybrid SPT heuristic algorithm is proposed. Chen et al. [38] used the concept hybrid to move from local optimal solutions to near-optimal solutions. Liu and MacCarthy [39] used to arrange jobs in a nondecreasing order of weight-factors of processing time and their corresponding ready time for jobs for the flow time criterion. Thus, we attempt to adopt a dynamic weight which is dependent on the job size due to learning effect. The steps of the proposed algorithm are described as follows.

The details of the first heuristic algorithm (HA) are provided as shown in Algorithm 1.

The complexities of each stage in the proposed heuristic algorithm are $O(n^2)$. Therefore, the complexity for the proposed heuristic algorithm is $O((k+1)n^2)$.

Input $J = \{J_1, J_2, \ldots, J_n\}$, n, and $k = [n/2]$;
  For $l \leftarrow 1$ **to** $k + 1$ **do**
    Set $\alpha = (l - 1)/t$
    For $i \leftarrow 1$ **to** n **do**
      Choose job J_u from J with $\min_{j \in J}\{\alpha p_j + (1 - \alpha) r_j\}$ to be placed job J_u in the ith position
    Enddo
enddo
Output $\text{sol}(S'_1), \text{sol}(S'_2), \ldots, \text{sol}(S'_{k+1})$ and $\min\{\text{sol}(S'_1), \text{sol}(S'_2), \ldots, \text{sol}(S'_{k+1})\}$.
End

Algorithm 1: HA algorithm.

4. Simulation Results

In this section we conduct a computational experiment to evaluate the performances of all proposed algorithms. All the algorithms were coded in Fortran and run on Compaq Visual Fortran version 6.6 on a 3.4 GHz Pentium 4 CPU with 1 GB RAM on Windows XP. Following Reeves [40] setting, we generated the job processing time from a uniform distribution $U(1, 20)$ and generated job ready times from another uniform distribution $U(0, 20n\lambda)$, where λ is taken as the values $1/n$, 0.25, 0.5, 0.75, and 1.

For the branch-and-bound algorithm, we reported the average and the maximum numbers of nodes and the average and the maximum execution time (in seconds). For the heuristic algorithms, we reported the mean and the maximum error percentages. The error is defined as

$$\frac{(H - O^*)}{O^*}, \tag{10}$$

where H and O^* are the solutions obtained from the heuristic algorithm and the branch-and-bound algorithm, respectively.

To evaluate performances of the proposed propositions and lower bounds, in the first part, we set the number of jobs to 10 and set the learning effect as the value of −0.2. The branch-and-bound algorithm with neither lower bounds nor dominance properties was used as the base, and the dominance properties were included one each time. The results were given in Figure 1. It can be observed that all the proposed propositions are useful in the branch-and-bound algorithm, especially Propositions 1, 6, and 7 and lb_1.

To investigate the impact of the control variable λ, in the second part, we set the number of jobs to 12, the learning effect a as the values of −0.05, −0.1, −0.15, and −0.2, and the control variable λ as the values of $1/n$ and from 0.2 to 1, with a jump of 0.05 each time. For each condition, 1000 instances were randomly generated. The results were summarized in Figures 2 and 3. It can be observed in Figure 2 that when the learning effect is fixed, the average number of nodes in the branch-and-bound algorithm first increases and then decreases as λ increases. It can be observed that Proposition 7 is useful in that case since the job completion time for

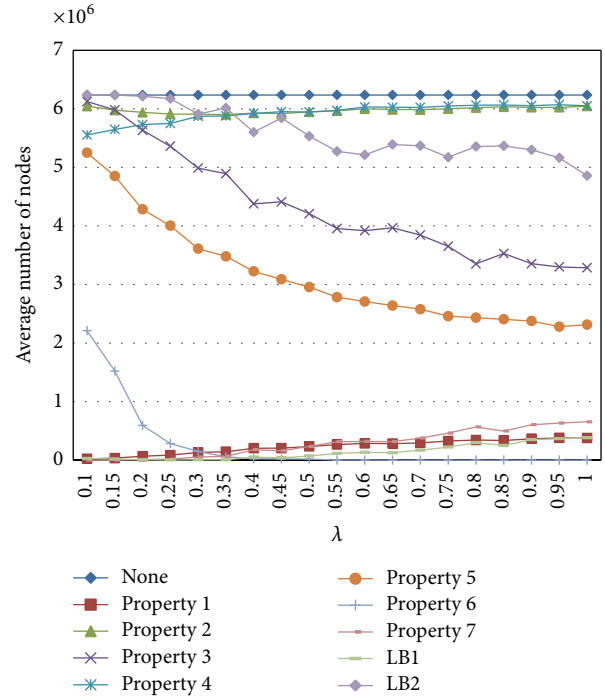

Figure 1: The performances of dominances and lower bound.

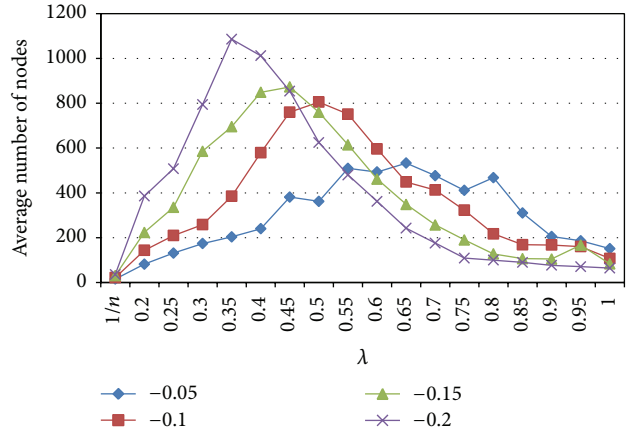

Figure 2: The performance of branch-and-bound algorithm over λ and a.

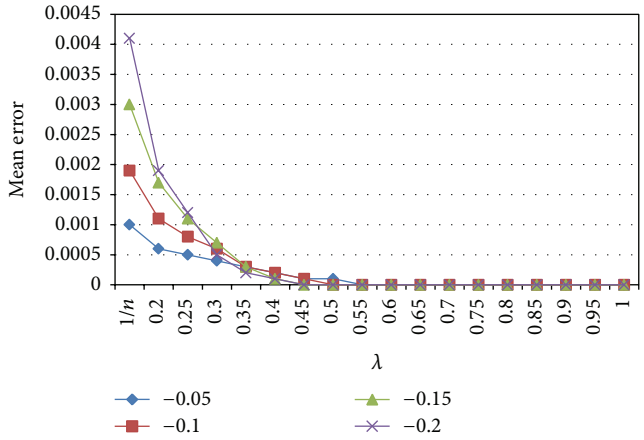

FIGURE 3: The performance of heuristic algorithm over λ and a.

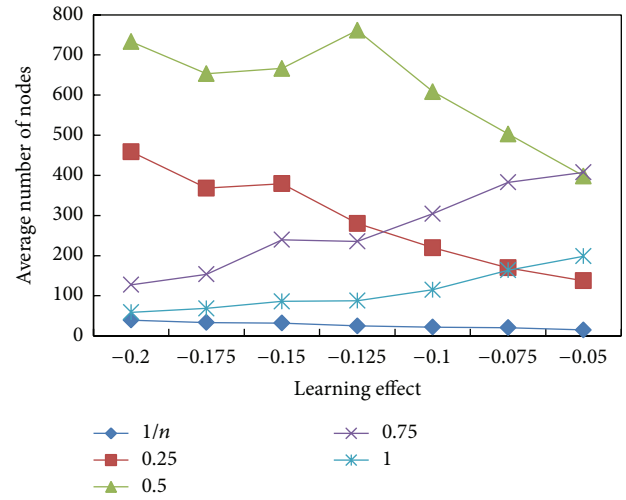

FIGURE 4: The performance of branch-and-bound algorithm over different learning effect.

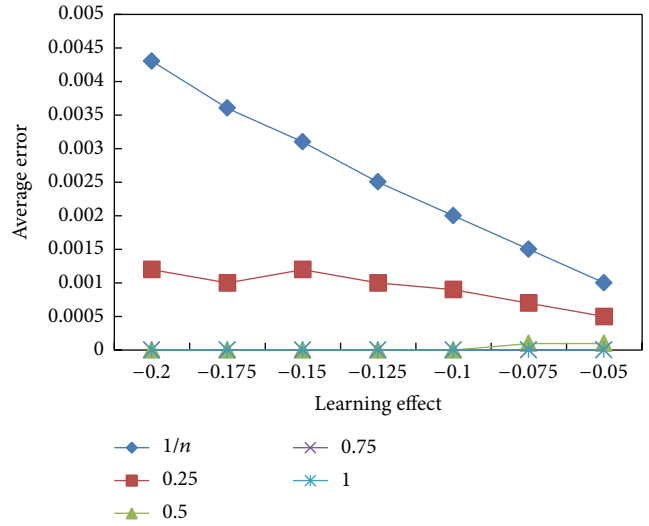

FIGURE 5: The performance of heuristic algorithm over different learning effect.

the partial scheduled jobs is easily larger than the maximal ready time of the unscheduled jobs in small value λ case. On the other hand, when the value of λ becomes larger, Proposition 6 becomes useful in that case. The similar behavior can be seen in Figure 3. The mean error decreased to zero as the value of λ became larger.

To evaluate the impact of learning effect, in the third part we set the job size to 12, the learning effect to from -0.2 to -0.05 with a jump of 0.025 each time, and λ to the values of $1/n$, 0.25, 0.5, 0.75, and 1. We test a total of 35 cases for this part. We generated 1000 instances for each case and summarized the results in Figures 4 and 5. Figure 4 indicated that the mean number of nodes increases as the learning effect becomes strong when $\lambda = 1/n$, 0.25, and 0.5. As the learning effect is stronger, Proposition 7 is less efficient since it is more difficult for the job completion time to surpass the maximal job ready time. Meanwhile, the mean number of nodes decreases as the learning effect becomes stronger when $\lambda = 0.75$ and 1. When the value of λ becomes large, Proposition 6 becomes useful. Figure 5 showed that the performance of HA algorithm performs very well with average error of less than 0.0045.

To evaluate the performances of all proposed algorithms over different parameters, in the fourth part we set the number of jobs to 12, 16, 20, and 24, the control variable θ to the values of $1/n$, 0.25, 0.5, 0.75, and 1, and the learning effect a to the values of -0.05, -0.1, -0.15, and -0.2 (note that if the learning effect took values less than -0.2, then the job processing time would approach zero very fast. Therefore, it was set to take values no less than -0.2). We generated 100 instances randomly for each case and summarized all the results in Table 1.

As shown in Table 1, regardless of n or learning rate variations, all the instances at $\lambda = 1/n$ are easily found out to be an optimal solution since Proposition 7 is powerful. On the other hand, the instances easily are solved out when the value of λ is getting larger due to Proposition 6. It also can be seen that the number of nodes becomes exponentially as the number of jobs increases larger. When fixed $n = 24$, there are 11 cases in which the branch-and-bound algorithm took larger than 10^8 nodes to solve out the instances. It can be found that the worst performance is located at case $(n, a, \lambda) = (24, -0.1, 0.25)$ with 9.9×10^8 nodes and 8322 seconds. Furthermore, Figure 6 indicated that there is no clear trend. Figure 7 further showed that the branch-and-bound algorithm can solve most of the problems in a reasonable amount of time when the job size is less than or equal to 24, excluding some difficult instances. For the performance of the proposed heuristic algorithm, out of the 80 evaluations, Table 1 indicated that all mean error percentages are less than 0.5%. Even in all difficult cases at $\lambda = 0.25$ the performance of proposed algorithm was not affected. Moreover, HA also was not affected as the

Table 1: The performance of the branch-and-bound and the heuristic algorithms.

n	a	λ	Branch-and-bound algorithm				HA	
			Node		CPU time (sec.)		Error	
			Mean	Max	Mean	Max	Mean	Max
12	−0.05	$1/n$	17	123	0.0008	0.0156	0.0017	0.0176
		0.25	158	1672	0.0048	0.0047	0.0012	0.0187
		0.50	374	4711	0.0058	0.0625	0.0010	0.0172
		0.75	666	17380	0.0092	0.2500	0.0001	0.0053
		1.00	213	7429	0.0033	0.0938	0.0001	0.0048
	−0.10	$1/n$	9	55	0.0009	0.0156	0.0022	0.0079
		0.25	244	2207	0.0053	0.0312	0.0014	0.0121
		0.50	881	14373	0.0116	0.1562	0.0005	0.0142
		0.75	486	17778	0.0062	0.1875	0.0003	0.0119
		1.00	102	1447	0.0020	0.0312	0.0002	0.0083
	−0.15	$1/n$	10	119	0.0011	0.0156	0.0030	0.0125
		0.25	346	3263	0.0070	0.0469	0.0019	0.0387
		0.50	1169	23592	0.0148	0.2656	0.0005	0.0186
		0.75	180	3484	0.0025	0.0312	0.0002	0.0096
		1.00	70	841	0.0008	0.0156	0.0003	0.0129
	−0.20	$1/n$	13	224	0.0019	0.0156	0.0004	0.0151
		0.25	678	8671	0.0125	0.1562	0.0022	0.0215
		0.50	440	5711	0.0059	0.0781	0.0008	0.0185
		0.75	110	1852	0.0017	0.0312	0.0005	0.0121
		1.00	48	269	0.0014	0.0156	0.0000	0.0045
16	−0.05	$1/n$	48	719	0.0066	0.0781	0.0009	0.0039
		0.25	5431	131044	0.3069	5.5469	0.0006	0.0086
		0.50	38532	1411457	0.8681	31.8281	0.0008	0.0116
		0.75	15999	1107504	0.3728	25.7812	0.0001	0.0040
		1.00	870	16849	0.0211	0.3750	0.0001	0.0046
	−0.10	$1/n$	64	673	0.0081	0.0781	0.0022	0.0083
		0.25	13367	859718	0.5369	29.1875	0.0012	0.0184
		0.50	21592	535276	0.4861	10.6562	0.0002	0.0118
		0.75	1925	54132	0.0526	1.5469	0.0003	0.0094
		1.00	261	2075	0.0076	0.0469	0.0001	0.0033
	−0.15	$1/n$	66	369	0.0078	0.0469	0.0038	0.0180
		0.25	19822	712421	0.6286	14.0938	0.0010	0.0092
		0.50	54197	2918302	1.1222	53.7500	0.0002	0.0066
		0.75	1801	109046	0.0478	2.9531	0.0001	0.0050
		1.00	197	2892	0.0062	0.0781	0.0000	0.0017
	−0.20	$1/n$	171	4282	0.0184	0.3906	0.0038	0.0107
		0.25	27159	725798	0.8030	16.3594	0.0015	0.0208
		0.50	10999	712605	0.2536	16.3750	0.0005	0.0127
		0.75	320	4508	0.0087	0.1406	0.0000	0.0040
		1.00	109	2101	0.0031	0.0625	0.0001	0.0068
	−0.05	$1/n$	158	1761	0.0253	0.2812	0.0009	0.0041
		0.25	337842	7781036	23.2809	468.5000	0.0007	0.0062
		0.50	1676257	59366260	39.0003	1204.8750	0.0001	0.0048
		0.75	27610	1036801	0.7211	26.7188	0.0001	0.0033
		1.00	3704	182198	0.0926	4.0625	0.0000	0.0026

Table 1: Continued.

n	a	λ	Branch-and-bound algorithm				HA	
			Node		CPU time (sec.)		Error	
			Mean	Max	Mean	Max	Mean	Max
20	−0.10	$1/n$	147	1100	0.0265	0.1875	0.0020	0.0064
		0.25	525486	18272556	29.7661	931.7031	0.0009	0.0076
		0.50	1104005	58152896	25.7080	1319.7812	0.0001	0.0049
		0.75	4288	121994	0.1178	3.4219	0.0000	0.0033
		1.00	1286	24316	0.0040	0.8594	0.0000	0.0000
	−0.15	$1/n$	357	5056	0.0590	0.7500	0.0027	0.0127
		0.25	1394787	35061512	54.7898	1016.5000	0.0010	0.0096
		0.50	98518	3076627	2.7577	100.0469	0.0000	0.0030
		0.75	5682	391725	0.1486	9.4219	0.0000	0.0000
		1.00	1104	76371	0.0314	2.1250	0.0001	0.0028
	−0.20	$1/n$	458	8952	0.0720	1.1719	0.0040	0.0105
		0.25	2498437	44522213	77.4317	1213.4531	0.0012	0.0095
		0.50	11662	493749	0.3208	12.7500	0.0002	0.0052
		0.75	543	5827	0.0161	0.1406	0.0001	0.0035
		1.00	317	5588	0.0097	0.1562	0.0000	0.0000
24	−0.05	$1/n$	350	9729	0.0876	2.1250	0.0009	0.0036
		0.25	8305284	76961550	829.5079	6601.4219	0.0004	0.0046
		0.50	6064502	66135367	216.9024	2553.3438	0.0002	0.0044
		0.75	430956	16808123	13.7014	507.8281	0.0000	0.0000
		1.00	14553	288343	0.4666	6.4531	0.0000	0.0000
	−0.10	$1/n$	616	9924	0.1622	2.3750	0.0017	0.0046
		0.25	16901968	99754077	1432.9100	8322.3906	0.0008	0.0078
		0.50	3412720	78241619	119.5013	2819.4062	0.0000	0.0043
		0.75	173689	11582237	4.8338	279.3125	0.0000	0.0028
		1.00	2911	82129	0.1058	2.7500	0.0000	0.0000
	−0.15	$1/n$	562	6245	0.1494	1.5312	0.0027	0.0102
		0.25	9840876	79790848	558.4099	7802.9844	0.0006	0.0040
		0.50	370863	13626535	13.3567	479.3594	0.0000	0.0047
		0.75	3971	53339	0.1498	2.1406	0.0000	0.0000
		1.00	2929	194770	0.1219	8.4688	0.0000	0.0000
	−0.20	$1/n$	2043	67463	0.5033	15.7969	0.0044	0.0122
		0.25	10286808	99187625	504.8521	7724.5156	0.0005	0.0076
		0.50	30923	562049	0.9963	14.6250	0.0000	0.0042
		0.75	1685	28692	0.0706	1.0625	0.0001	0.0029
		1.00	431	5241	0.0184	0.1719	0.0000	0.0021

values of learning rate or release time vary because all of the maximum values of the worst cases of the algorithms were less than 5%. Thus, HA algorithm is recommended due to its accuracy.

5. Conclusions

In this paper we studied a single-machine scheduling problem to minimize the makespan. We considered job processing times as the decreasing functions of their already processed jobs and all the jobs have their different ready times. Due to the fact that the same problem without a learning effect has been NP-hard one, we then proposed a branch-and-bound algorithm and a heuristic algorithm for this problem.

The results showed that the branch-and-bound algorithm can solve the instances up to $n = 24$, and the proposed heuristic HA is very accurate.

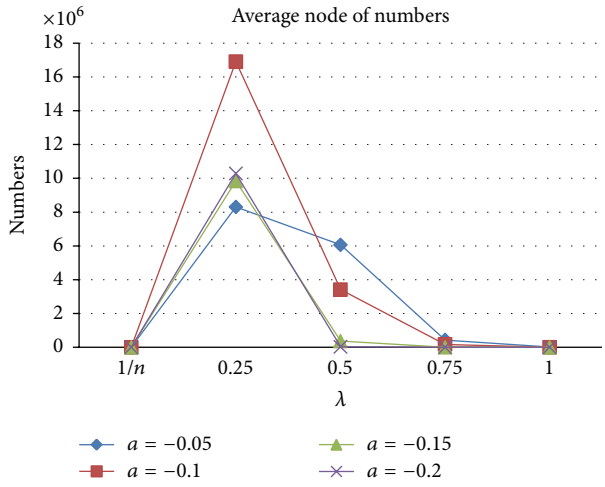

Figure 6: The behaviors of the nodes of the branch-and-bound algorithm at $n = 24$.

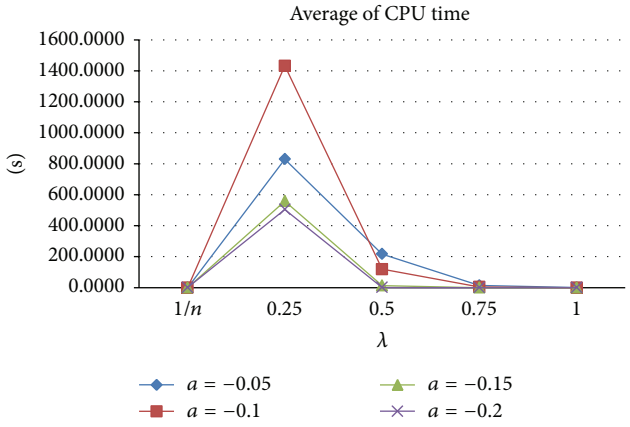

Figure 7: The performance of CPU times of the branch-and-bound algorithm at $n = 24$.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are grateful to the editor and two anonymous referees for their constructive comments on the earlier version of their paper. This paper was supported in part by the Ministry of Science and Technology of Taiwan under grant number NSC 102-2221-E-035-070-MY3; MOST 103-2410-H-035-022-MY2.

References

[1] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173–178, 1999.

[2] T. C. E. Cheng and G. Wang, "Single machine scheduling with learning effect considerations," *Annals of Operations Research*, vol. 98, pp. 273–290, 2000.

[3] D. Biskup, "A state-of-the-art review on scheduling with learning effects," *European Journal of Operational Research*, vol. 188, no. 2, pp. 315–329, 2008.

[4] T. C. E. Cheng, C.-C. Wu, and W.-C. Lee, "Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects," *Information Sciences*, vol. 178, no. 11, pp. 2476–2487, 2008.

[5] T. Eren and E. Güner, "A bicriteria flowshop scheduling with a learning effect," *Applied Mathematical Modelling*, vol. 32, no. 9, pp. 1719–1733, 2008.

[6] T. Eren and E. Güner, "A bicriteria parallel machine scheduling with a learning effect," *International Journal of Advanced Manufacturing Technology*, vol. 40, no. 11-12, pp. 1202–1205, 2009.

[7] T. Eren, "A note on minimizing maximum lateness in an m-machine scheduling problem with a learning effect," *Applied Mathematics and Computation*, vol. 209, no. 2, pp. 186–190, 2009.

[8] A. Janiak and R. Rudek, "A note on the learning effect in multi-agent optimization," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5974–5980, 2011.

[9] M. D. Toksarı, D. Oron, and E. Güner, "Single machine scheduling problems under the effects of nonlinear deterioration and time-dependent learning," *Mathematical and Computer Modelling*, vol. 50, no. 3-4, pp. 401–406, 2009.

[10] M. D. Toksari and E. Güner, "The common due-date early/tardy scheduling problem on a parallelmachine under the effects of time-dependent learning and linear and nonlinear deterioration," *Expert Systems with Applications*, vol. 37, no. 1, pp. 92–112, 2010.

[11] J.-B. Wang and L.-L. Liu, "Two-machine flow shop problem with effects of deterioration and learning," *Computers & Industrial Engineering*, vol. 57, no. 3, pp. 1114–1121, 2009.

[12] J.-B. Wang, C. T. Ng, T. C. E. Cheng, and L. L. Liu, "Single-machine scheduling with a time-dependent learning effect," *International Journal of Production Economics*, vol. 111, no. 2, pp. 802–811, 2008.

[13] Y. Yin, D. Xu, K. Sun, and H. Li, "Some scheduling problems with general position-dependent and time-dependent learning effects," *Information Sciences*, vol. 179, no. 14, pp. 2416–2425, 2009.

[14] Y. Yin, D. Xu, and J. Wang, "Some single-machine scheduling problems with past-sequence-dependent setup times and a general learning effect," *The International Journal of Advanced Manufacturing Technology*, vol. 48, no. 9–12, pp. 1123–1132, 2010.

[15] Y. Yin, D. Xu, and J. Wang, "Single-machine scheduling with a general sum-of-actual-processing-times-based and job-position-based learning effect," *Applied Mathematical Modelling*, vol. 34, no. 11, pp. 3623–3630, 2010.

[16] Y. Yin and D. Xu, "Some single-machine scheduling problems with general effects of learning and deterioration," *Computers and Mathematics with Applications*, vol. 61, no. 1, pp. 100–108, 2011.

[17] Y. Yin, M. Liu, J. Hao, and M. Zhou, "Single-machine scheduling with job-position-dependent learning and time-dependent deterioration," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 42, no. 1, pp. 192–200, 2012.

[18] C.-C. Wu, Y. Yin, and S.-R. Cheng, "Some single-machine scheduling problems with a truncation learning effect," *Computers & Industrial Engineering*, vol. 60, no. 4, pp. 790–795, 2011.

[19] C.-C. Wu, Y. Yin, W.-H. Wu, and S.-R. Cheng, "Some polynomial solvable single-machine scheduling problems with a truncation sum-of-processing-times based learning effect," *European Journal of Industrial Engineering*, vol. 6, no. 4, pp. 441–453, 2012.

[20] C.-C. Wu, Y. Yin, and S.-R. Cheng, "Single-machine and two-machine flowshop scheduling problems with truncated position-based learning functions," *Journal of the Operational Research Society*, vol. 64, no. 1, pp. 147–156, 2013.

[21] J.-B. Wang, M.-Z. Wang, and P. Ji, "Single machine total completion time minimization scheduling with a time-dependent learning effect and deteriorating jobs," *International Journal of Systems Science*, vol. 43, no. 5, pp. 861–868, 2012.

[22] J.-B. Wang, Y.-B. Wu, and P. Ji, "A revision of some single-machine and m-machine flowshop scheduling problems with learning considerations," *Information Sciences*, vol. 190, pp. 227–232, 2012.

[23] J.-B. Wang, X.-Y. Wang, L.-H. Sun, and L.-Y. Sun, "Scheduling jobs with truncated exponential learning functions," *Optimization Letters*, vol. 7, no. 8, pp. 1857–1873, 2013.

[24] J. Bai, M.-Z. Wang, and J.-B. Wang, "Single machine scheduling with a general exponential learning effect," *Applied Mathematical Modelling*, vol. 36, no. 2, pp. 829–835, 2012.

[25] B. Vahedi-Nouri, P. Fattahi, and R. Ramezanian, "Minimizing total flow time for the non-permutation flow shop scheduling problem with learning effects and availability constraints," *Journal of Manufacturing Systems*, vol. 32, no. 1, pp. 167–173, 2013.

[26] Y.-Y. Lu, G. Li, Y.-B. Wu, and P. Ji, "Optimal due-date assignment problem with learning effect and resource-dependent processing times," *Optimization Letters*, vol. 8, no. 1, pp. 113–127, 2014.

[27] A. Bachman and A. Janiak, "Scheduling jobs with position-dependentprocessing times," *Journal of the Operational Research Society*, vol. 55, no. 3, pp. 257–264, 2004.

[28] W.-C. Lee, C.-C. Wu, and M.-F. Liu, "A single-machine bi-criterion learning scheduling problem with release times," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10295–10303, 2009.

[29] T. Eren, "Minimizing the total weighted completion time on a single machine scheduling with release dates and a learning effect," *Applied Mathematics and Computation*, vol. 208, no. 2, pp. 355–358, 2009.

[30] W.-C. Lee, C.-C. Wu, and P.-H. Hsu, "A single-machine learning effect scheduling problem with release times," *Omega*, vol. 38, no. 1-2, pp. 3–11, 2010.

[31] C.-C. Wu and C.-L. Liu, "Minimizing the makespan on a single machine with learning and unequal release times," *Computers & Industrial Engineering*, vol. 59, no. 3, pp. 419–424, 2010.

[32] C. C. Wu, P. H. Hsu, and K. Lai, "Simulated-annealing heuristics for the single-machine scheduling problem with learning and unequal job release times," *Journal of Manufacturing Systems*, vol. 30, no. 1, pp. 54–62, 2011.

[33] C.-C. Wu, P.-H. Hsu, J.-C. Chen, N.-S. Wang, and W.-H. Wu, "Branch-and-bound and simulated annealing algorithms for a total weighted completion time scheduling with ready times and learning effect," *International Journal of Advanced Manufacturing Technology*, vol. 55, no. 1–4, pp. 341–353, 2011.

[34] C.-C. Wu, P.-H. Hsu, J.-C. Chen, and N.-S. Wang, "Genetic algorithm for minimizing the total weighted completion time scheduling problem with learning and release times," *Computers and Operations Research*, vol. 38, no. 7, pp. 1025–1034, 2011.

[35] J.-B. Wang and J.-J. Wang, "Single machine group scheduling with time dependent processing times and ready times," *Information Sciences*, vol. 275, pp. 226–231, 2014.

[36] M. M. Dessouky, "Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness," *Computers and Industrial Engineering*, vol. 34, no. 4, pp. 793–806, 1998.

[37] W.-H. Kuo and D.-L. Yang, "Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect," *European Journal of Operational Research*, vol. 174, no. 2, pp. 1184–1190, 2006.

[38] J.-S. Chen, J. C.-H. Pan, and C.-M. Lin, "Solving the reentrant permutation flow-shop scheduling problem with a hybrid genetic algorithm," *International Journal of Industrial Engineering : Theory Applications and Practice*, vol. 16, no. 1, pp. 23–31, 2009.

[39] J. Liu and B. L. MacCarthy, "Effective heuristics for the single machine sequencing problem with ready times," *International Journal of Production Research*, vol. 29, no. 8, pp. 1521–1533, 1991.

[40] C. Reeves, "Heuristics for scheduling a single machine subject to unequal job release times," *European Journal of Operational Research*, vol. 80, no. 2, pp. 397–403, 1995.